For example, if the URL specifies a static content resource 122, such as an HTML file, a handler 120 accesses the static content resource 122 and passes the static content resource 122 back through the HTTP pipeline 118 for communication to the client 100 in an HTTP response 112. Alternatively, in an embodiment of the present invention, if the URL specifies a dynamic content resource 124, such as an ASP+ resource, a handler 120 accesses the dynamic content resource 124, processes the contents of the dynamic content resource 124, and generates the resulting HTML code for the web page 104. In an embodiment of the present invention, the resulting HTML code includes standard HTML 3.2 code. Generally, a dynamic content resource is a server–side declaration datastore (e.g., an ASP+ resource) that can be used to dynamically generate the authoring language code that describes a web page to be displayed on a client. The HTML code for the web page is then passed through the HTTP pipeline 118 for communication to the client 100 in an HTTP response 112.

During its processing, a handler 120 can also access libraries of pre–developed or third–party code to simplify the development effort. One such library is a server–side class control library 126, from which the handler 120 can instantiate server–side control objects for processing user interface elements and generating the resultant HTML data for display of a web page. In an embodiment of the present invention, one or more server–side control objects map to one or more user interface elements, visible or hidden, on the web page described in the dynamic content resource 124. More details on server–side control objects for processing client–side server interface elements are described in U.S. Patent Application No. 09/573,769 entitled SERVER–SIDE CONTROL OBJECTS FOR PROCESSING CLIENT–SIDE USER INTERFACE ELEMENTS ~~{docket number MS144237.1/40062.0046–US–01}~~, incorporated herein by reference for all it discloses and teaches, filed concurrently herewith, still pending, and assigned to the Assignee of the present invention.

provides instructions to a page compiler that reads the file 400, creates and invokes the appropriate server-side control objects, and ultimately combines the rendered HTML code for transmission to the client in an HTTP response.

The first line of the file 400 includes a directive in the format:

<%@ directive {attribute=value} %>

where *directive* may include without limitation "page", "cache", or "import". Directives are used by the page compiler when processing a dynamic content resource to determine such characteristics as buffering semantics, session state requirements, error handling schemes, scripting languages, transaction semantics, and import directions. Directives may be located anywhere within a page file. For more details about the page compiler, see U.S. Patent Application No. 09/573,768, entitled SERVER-SIDE CODE GENERATION FROM A DYNAMIC WEB PAGE CONTENT FILE ~~[docket number MS-144238.1/40062.0047-US-01]~~, still pending filed concurrently herewith and assigned to the Assignee of the present application.

02/07/05

In the second line, <html> is a standard HTML starting tag, which is passed through to the resulting HTML code as a literal (i.e., without additional processing to render the resulting HTML code). In HTML, the <html> indicates the beginning of the HTML file and is paired with the closing tag on line 21, </html>, which is also a literal.

A code declaration block is located at lines 3-10 of the file 400. Generally, server-side code declaration blocks define page object and control object member variables and methods that are executed on the server. In the format:

<script runat = "server" [language = "*language*"][src = "*externalfile*"]>

.....

</script>

where the language and src parameters are optional. In an embodiment of the present invention, code declaration blocks are defined using <script> tags that contain a "runat" attribute having a

operation 606 restores the same control object to its previous state, in part by loading the text string "JDoe" into the property value. In addition, whether the state of a given object is stored and restored is configurable.

In summary of one embodiment of the present invention, the state of one or more server-side control objects is "saved" after processing. The saved state information is transmitted to the client in a response. The client returns the saved state information to the server in a subsequent response. The server loads the state information a freshly instantiated server-side control object hierarchy, such that the state of the hierarchy is restored to its previous state. More details on state management can be found in U.S. Patent Application No. 09/574,144 entitled STATE MANAGEMENT OF SERVER SIDE CONTROL OBJECTS ~~[docket number MS144236.1/40062.0045-US-01]~~ now U.S. Patent 6,757,900, incorporated herein by reference for all that it discloses and teaches, filed concurrently herewith, and assigned to the Assignee of the present application.

eo 2/07/05

An alternative embodiment may maintain the state information on the server or at some other web location accessible by the server during the round trip from the server to the client, and then back to the server. After the client request is received by the server, this state information may be retrieved by the server and loaded into the appropriate server-side control object(s) in the control object hierarchy.

In operation 608, postback data received from the HTTP request is processed. Postback data may be included in the payload of the HTTP request in key-value pairs, in a hierarchical representation (e.g., XML), or in other data representations, such as RDF ("Resource Description Framework"). Operation 608 parses the payload to identify a unique identifier of a target server-side control object. The target server-side control object can be nested within an arbitrary hierarchy. If the identifier (e.g. "page1:list1:listrow2:text1") is found and the corresponding target server-side control object exists in the control object hierarchy, the received postback data is passed to the control object. For example, referring to FIG. 1, a unique identifier

Operation 612 resolves data binding relationships between the server–side control objects and one or more databases accessible by the server, thereby updating in control object properties with database values and/or updating database fields with values of control object properties. In an embodiment of the present invention, properties of server–side control objects may be associated (or data bound) to properties of a parent data binding container, such as a table in a server–side application database. During the data binding operation 612, the page framework may update a data bound control object property with the value of the corresponding parent data binding container property. In this manner, user interface elements on the web page of the next response accurately reflect updated property values, because the control object properties to which the user interface elements correspond have been automatically updated during the data binding operation 612. Likewise, control object properties can also be updated to the parent data binding container fields, thereby updating a server–side application database with postback input from a server–side control object. More details regarding data binding using server–side control objects can be found in U.S. Patent Application No. 09/573,656 entitled DATABINDING USING SERVER–SIDE CONTROL OBJECTS ~~[docket number MS144240.1/40062.50-US-61]~~, incorporated herein by reference for all that it discloses and teaches, filed concurrently herewith, and assigned to the Assignee of the present application.

now U.S. Patent 6,792,607

Operation 614 performs miscellaneous update operations that may be executed before the control object state is saved and the output is rendered. Operation 616 requests state information (i.e., viewstate) from one or more control objects in the control object hierarchy and stores the state information for insertion into a transportable state structure that is communicated to the client in the HTTP response payload. For example, a "grid" control object may save a current index page of a list of values so that the "grid" control object may be restored to this state after a subsequent HTTP request (i.e., in operation 606). As described above, the viewstate information represents the state of the control object hierarchy prior to any subsequent actions by the client.

the transportable state structure can be found in previously incorporated U.S. Patent Application No. 09/574,449 now U.S. Patent 6,757,900 entitled STATE MANAGEMENT OF SERVER-SIDE CONTROL OBJECTS.

Examining operation 810 examines the incoming postback data in the HTTP request to find and extract identifiers ("IDs") corresponding to server-side control objects in the control object hierarchy. An identifier is generally found as a key in a key-value pair in the form data section of the HTTP request. Locating operation 812, having found a server-side control object ID, attempts to locate the corresponding server-side control object within the hierarchy. Locating operation 812 is described in more detail with regard to FIG. 9. Having found the corresponding server-side control object, loading operation 814 passes the associated postback data into the located server-side object. If loading operation 814 results in a change to the data previously stored in the server-side control object (i.e., previous data), data change operation 816 indicates a data change associated with the server-side control object. In an embodiment of the present invention, the previous data stored by the server-side control object is compared with the postback data to be stored, and, if the postback data differs from the previous data, a data change is indicated and the previous data is replaced with the postback data in the server-side control object. Alternative methods of indicating a data change and replacing the previous data with the postback data are also contemplated in the scope of the present invention.

In an embodiment of the present invention, a data change indication is performed by recording in a list an identifier of the control object for which the data change was detected. The data change indications may stored in an array or linked list, although other storage configurations are contemplated within the scope of the present invention. Decision operation 818 directs processing to the examining operation 810 if more postback data is available. In this manner, the actual raising of the data change events is deferred until postback data for other control objects has been input and the data change indication for such control objects is recorded into the list. After no other postback data remains to be processed, decision